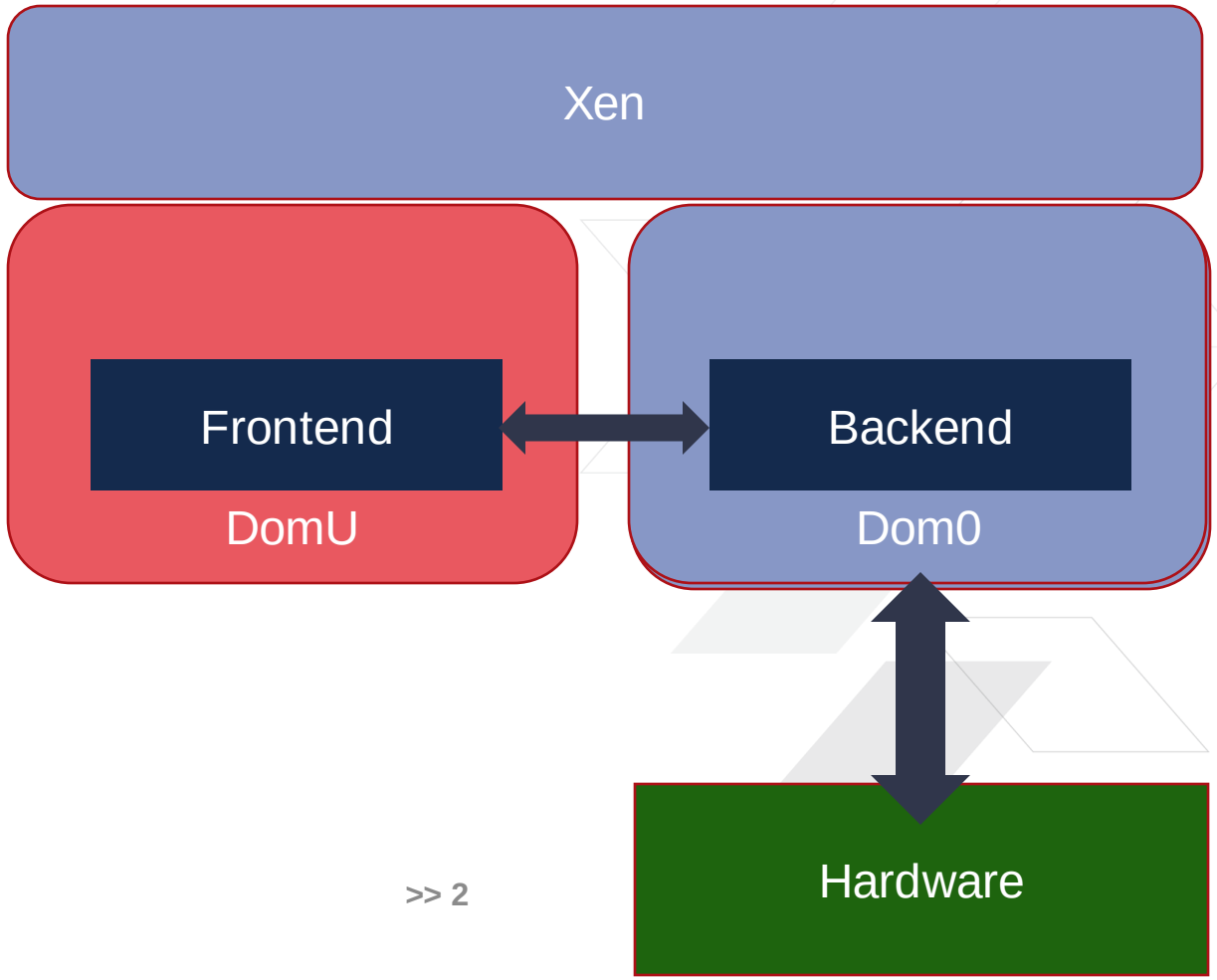
VM-to-VM Communication Mechanisms for Embedded

Stefano Stabellini

XILINX

Existing VM-to-VM Communication Mechanisms

- **Existing protocols: Xen PV Drivers, VirtIO**
  - discoverable and dynamic
  - create new connections at runtime
  - made for IO virtualization
- **A frontend in the guest connects to the backend in Dom0 / hypervisor**
- **Created to virtualize devices**
- **Typically based on memory sharing**
- **VirtIO expects privileged backends**

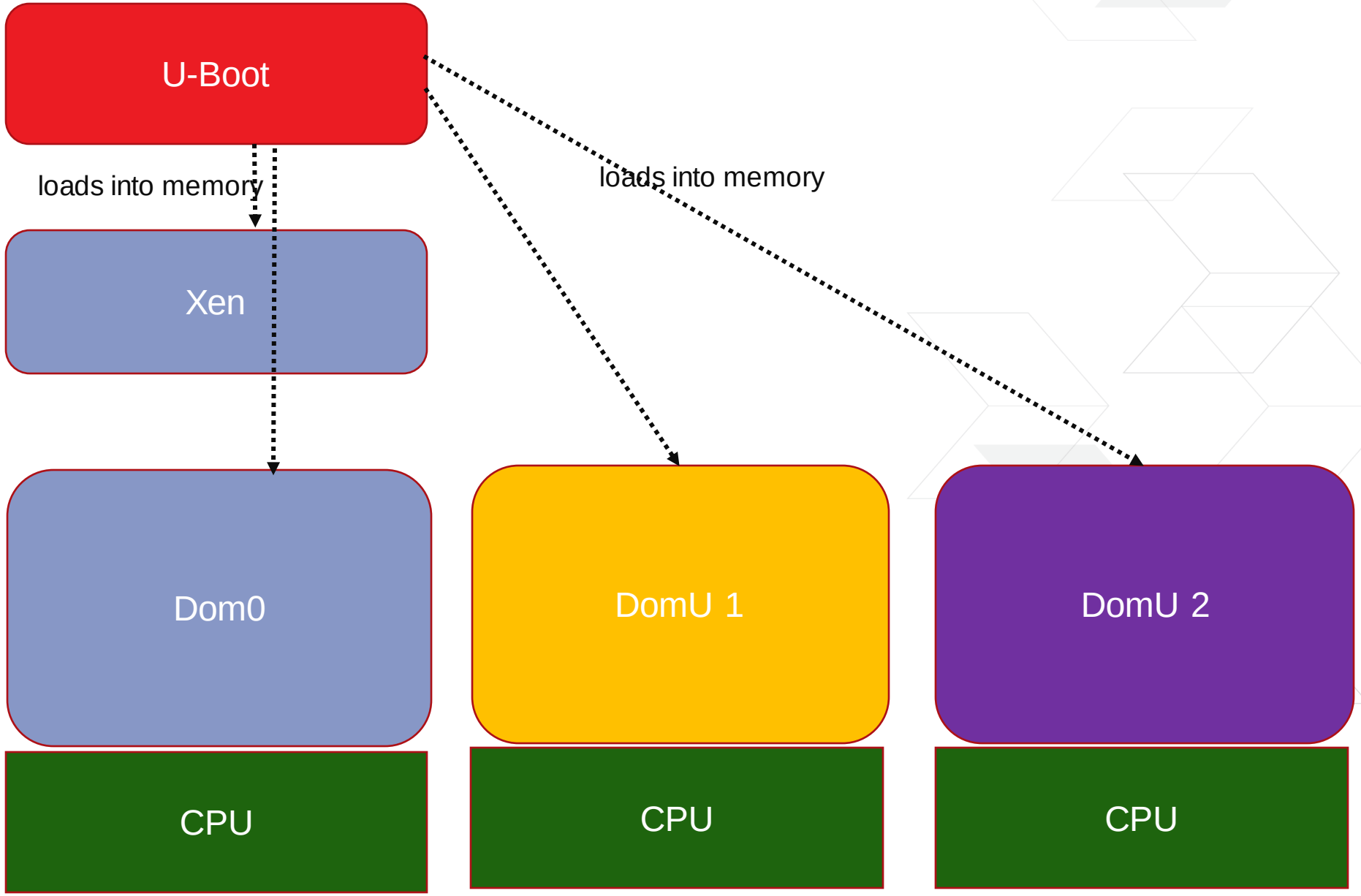


Static Partitioning

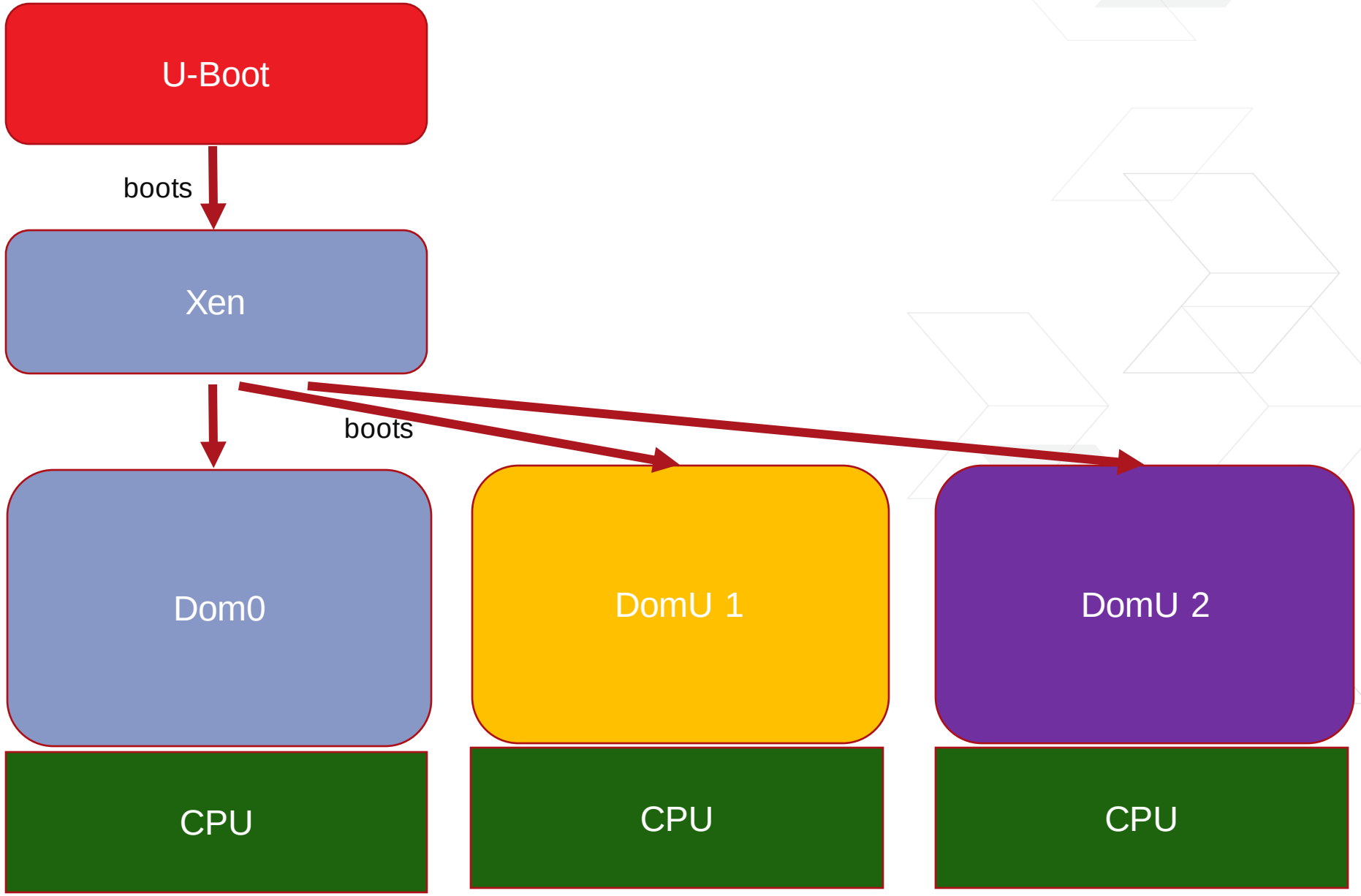
Static Partitioning

- **Static Partitioning is similar to virtualization with some key differences:**
  - No dynamic VMs, a limited number of "partitions" instead
  - Focus on direct assignment of hardware resources
  - Configuration defined at "Build Time"
  - Real-Time, Safety, and Short Boot Times are often key requirements
- **Example: Xen Dom0less**

Xen Dom0less



Xen Dom0less



Xen Dom0less Current Status

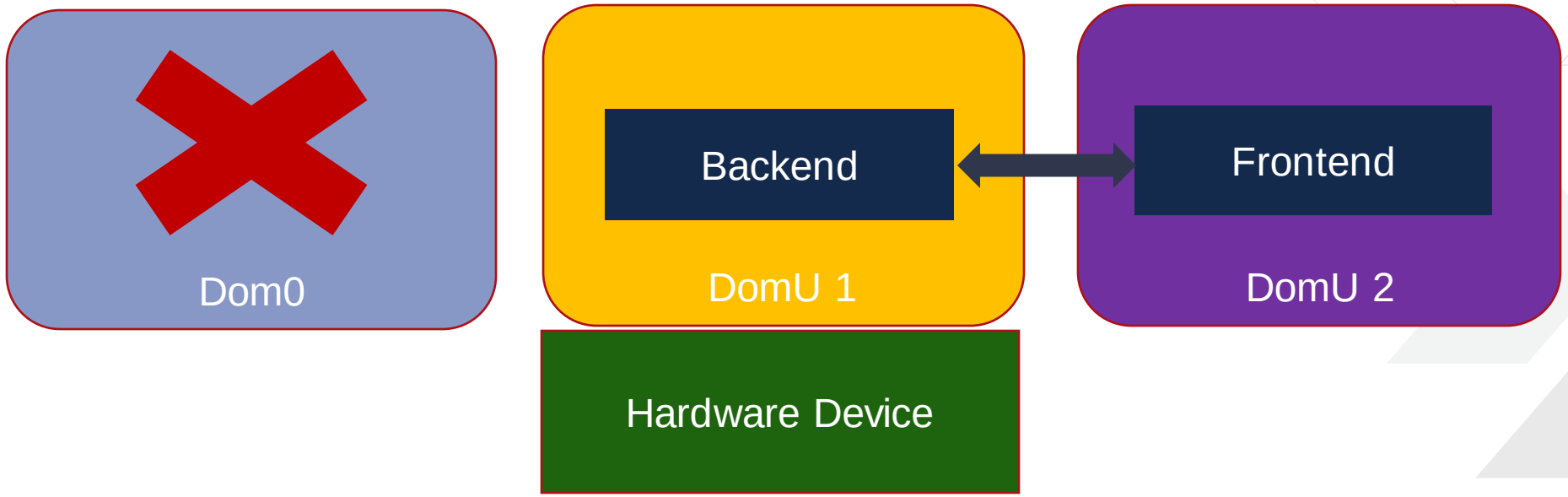
- Static Partitions defined at build time
- Fast boot times, real-time support, easier to safety-certify
- Dom0 is not required
- No "out of the box" communication mechanisms available

Static Partitioning and Communication

- **Often only VM-to-VM communication is required, not device virtualization**
  - There are enough physical devices to directly assign them to VMs as needed
  - Device virtualization can be interesting for sharing an SD card among multiple VMs
- **VM-to-VM communication is different from device virtualization**
  - A simple VM-to-VM channel to send and receive raw data
  - It doesn't need "frontends" and "backends"
  - It requires a smaller code base
  - It is faster for exchanging data but it is unwieldy for virtualizing devices
- **Static definition of VM-to-VM communication channels**
  - Define connections at "build time"
  - Required for safety
- **No privileged backends**
  - Required for safety
- **Support Linux and non-Linux guests (Zephyr, FreeRTOS, WindRiver, QNX, etc.)**

Static Partitioning VM-to-VM communication

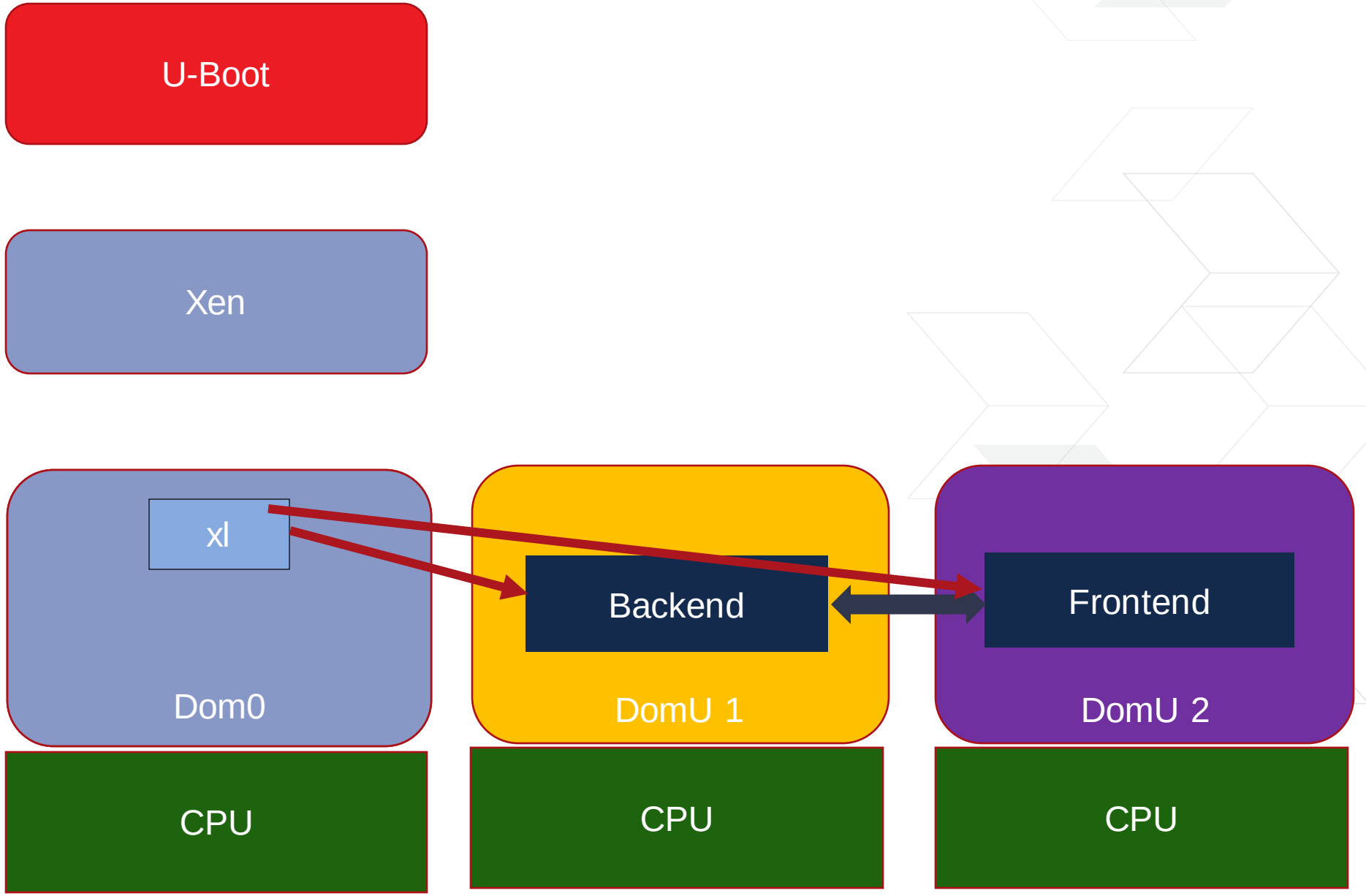
- **No privileged backends**



Xen PV Drivers

- **Solid and hardened in production for years (AWS)**
- **Made for device virtualization, can be used for communication:**
  - Network
  - Block (disks)
  - Console
  - 2D graphics, mouse and keyboard
  - Sound
  - Etc.
- **Pros:**
  - Very Fast Unprivileged Backends
  - Available for Linux, BSDs and Windows, less common among RTOSes
- **Cons:**
  - Might not be available in certain embedded RTOSes (but BSD versions exist for all PV drivers)
  - Dom0less support is work-in-progress

Xen PV Drivers and Dom0less



Xen PV Drivers and Dom0less

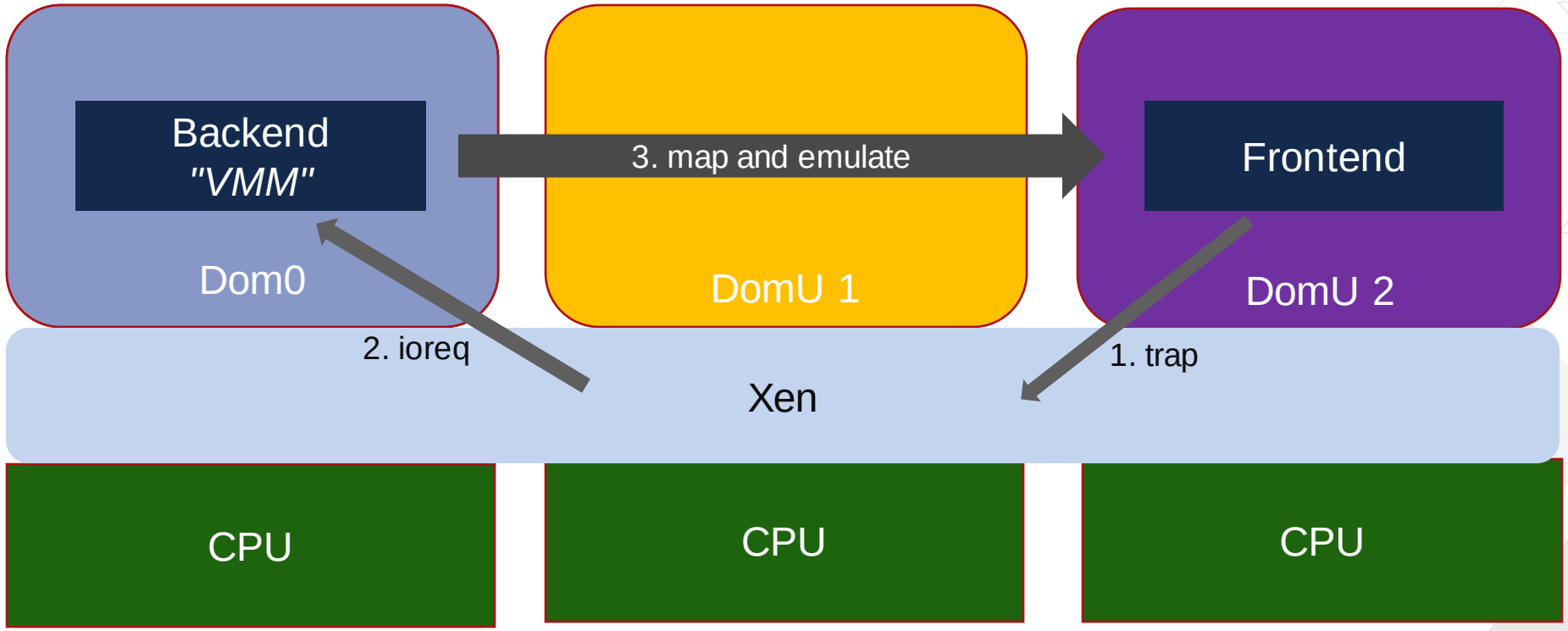
- **Domains booted in parallel**
- **PV Drivers connections created after Dom0 is up and running**
- **Advantages compared to regular non-Dom0less deployments:**
  - Domains are still started very quickly
  - Domains can immediately begin to perform critical tasks
  - Overall time to get PV Drivers up and running is shorter (no domain creation needed in Dom0)
- **To become available by the end of the year (work by Hipert/Lab @ Unimore)**

VirtIO

- **Frontend Drivers are available in most Operating Systems**
- **"VMM" provides the backends (e.g. QEMU, kvmtools, etc.)**

- **Pros:**
  - Many virtual device classes
- **Cons:**
  - Backends are currently required to be privileged – backends must be in Dom0
    - Security implications
    - Safety implications
  - Support for Xen is available, but it requires non-upstream toolstack patches
  - No Dom0less support

VirtIO

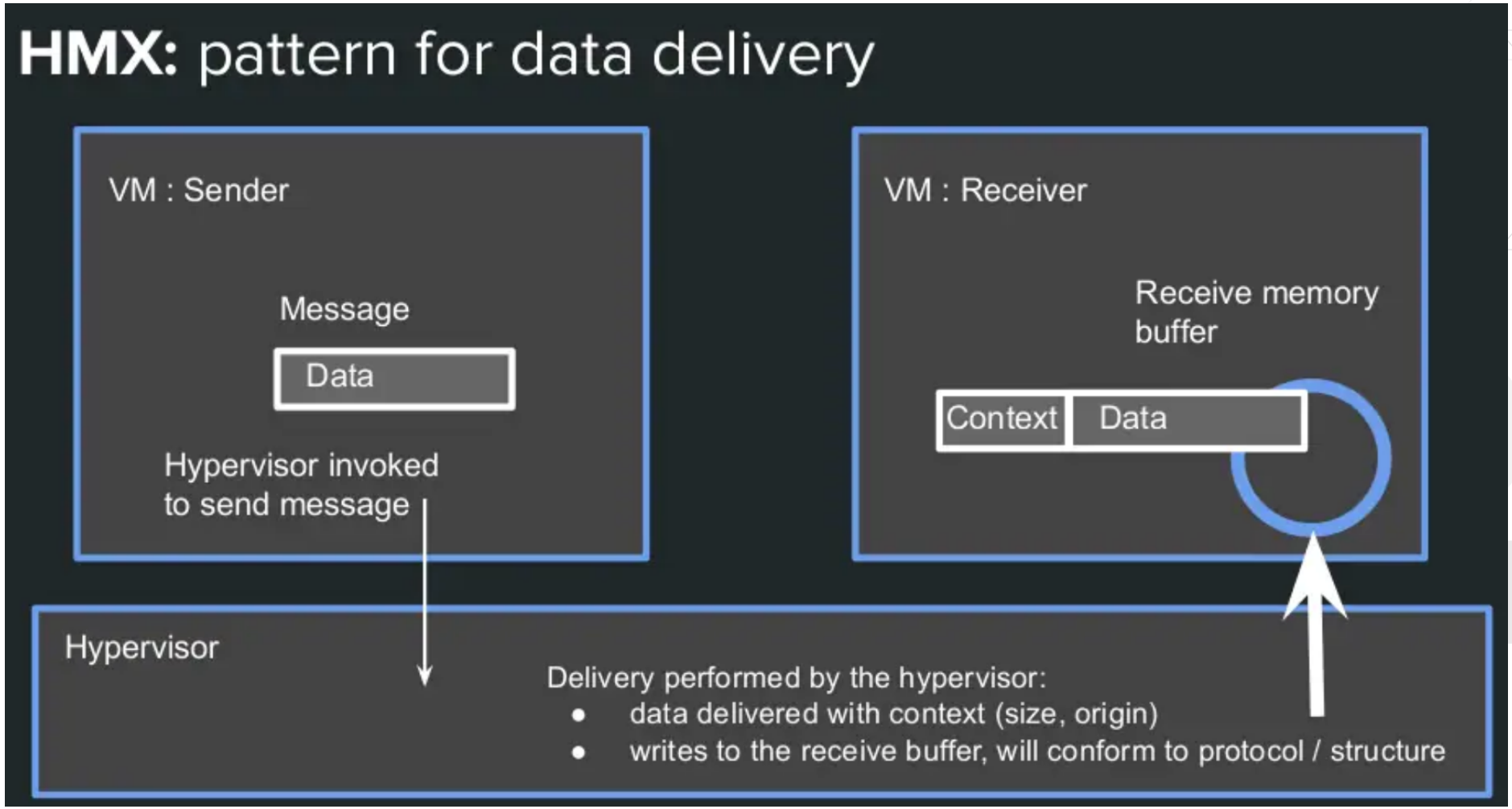

Backend
"VMM"
Dom0
3. map and emulate
DomU 1
Frontend
DomU 2
2. ioreq
1. trap
Xen
CPU
CPU
CPU

VirtIO

- **IOREQ infrastructure upstream in Xen**
  - It enables VirtIO backends and any other emulators to run in Dom0 (e.g. QEMU)
  - No support in the Xen tools for creating VirtIO frontends/backends yet (patch available)
  - PoC with virtio-block by EPAM
  - Requires Backends with full privileges, they have to be in Dom0
  - Good performance with full privileges
- **Support for Unprivileged Backend is work-in-progress by Linaro Project Stratos**
  - Based on memory copies to/from a pre-shared memory region
  - Performance to be determined (never done before, underlying protocols designed for sharing)
- **How to enable VirtIO for Dom0less?**
  - Could VirtIO device hotplug be used to avoid synchronous waiting during boot?

Argo

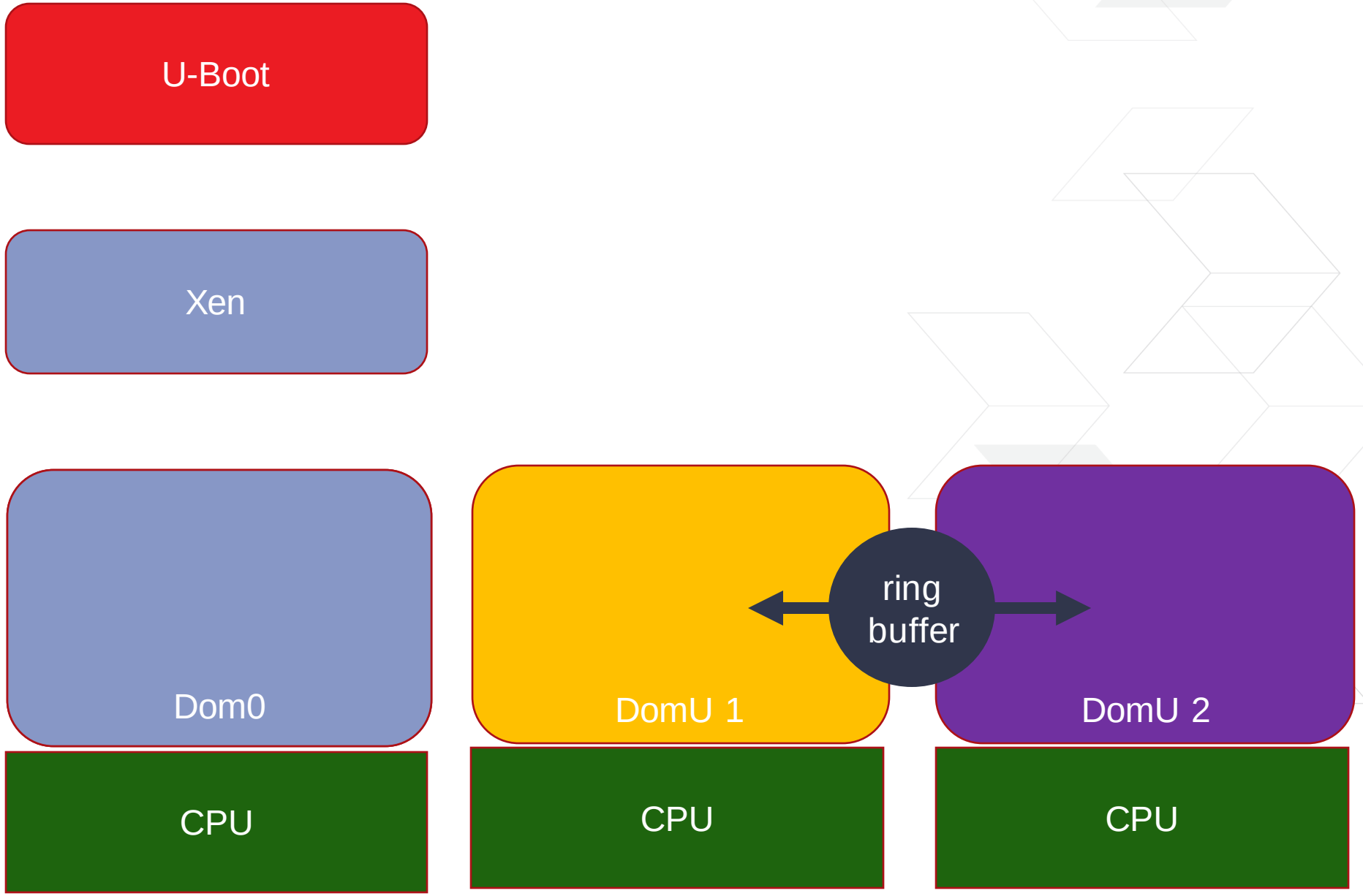
> **Hypervisor-Mediated Data Transfers**



Argo

- **Pros:**
  - Great performance and very strong security properties
    - Hypervisor checks against malicious data senders
    - Designed and optimized for memory copies
  - More lightweight than Xen PV Drivers and VirtIO
    - No Xenstore, no PV backends, no VMM needed
    - Requires Event Channels and Argo drivers (BSD drivers available here and here)
  - Straightforward Dom0less enablement: no need for any kind of "wait"
    - No need to wait for Dom0 to complete booting to communicate with other VMs
- **Current Status:**
  - It requires one Linux patch to work with Dom0less
    - Thanks Alec Kwapis from DornerWorks!
- **Cons:**
  - Requires Argo driver and Xen event channels for notifications

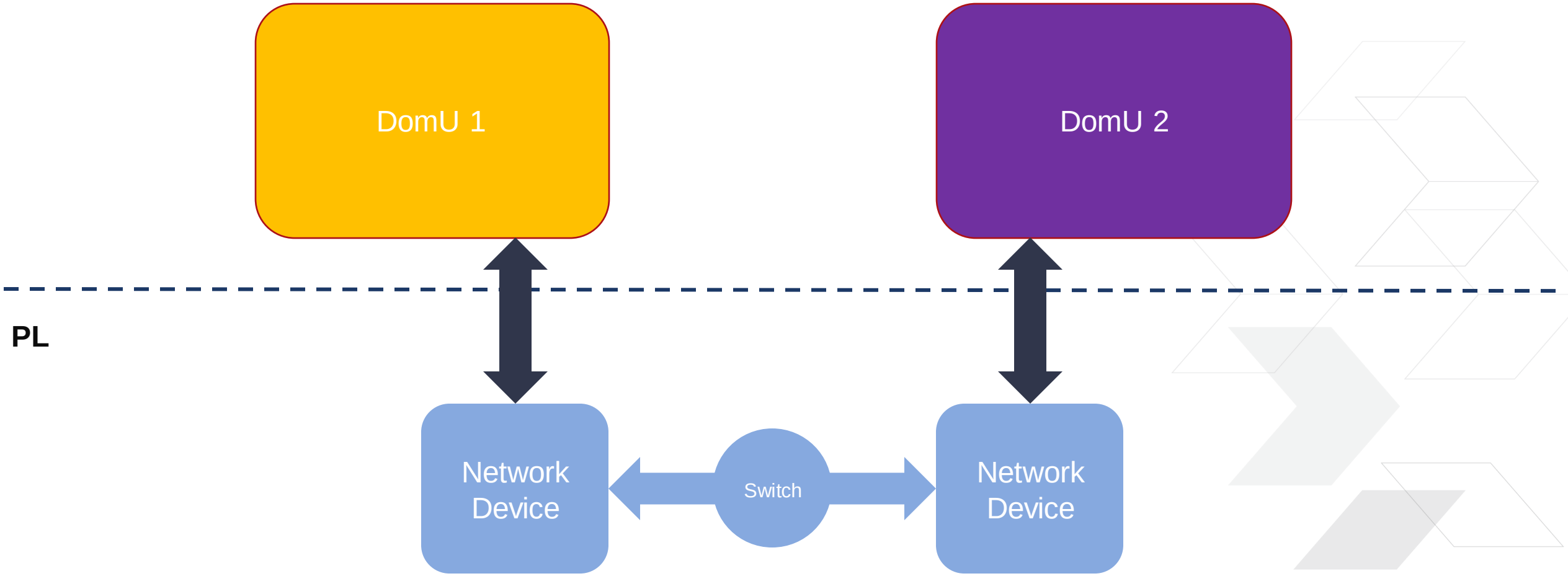
Static Shared Memory and Interrupts



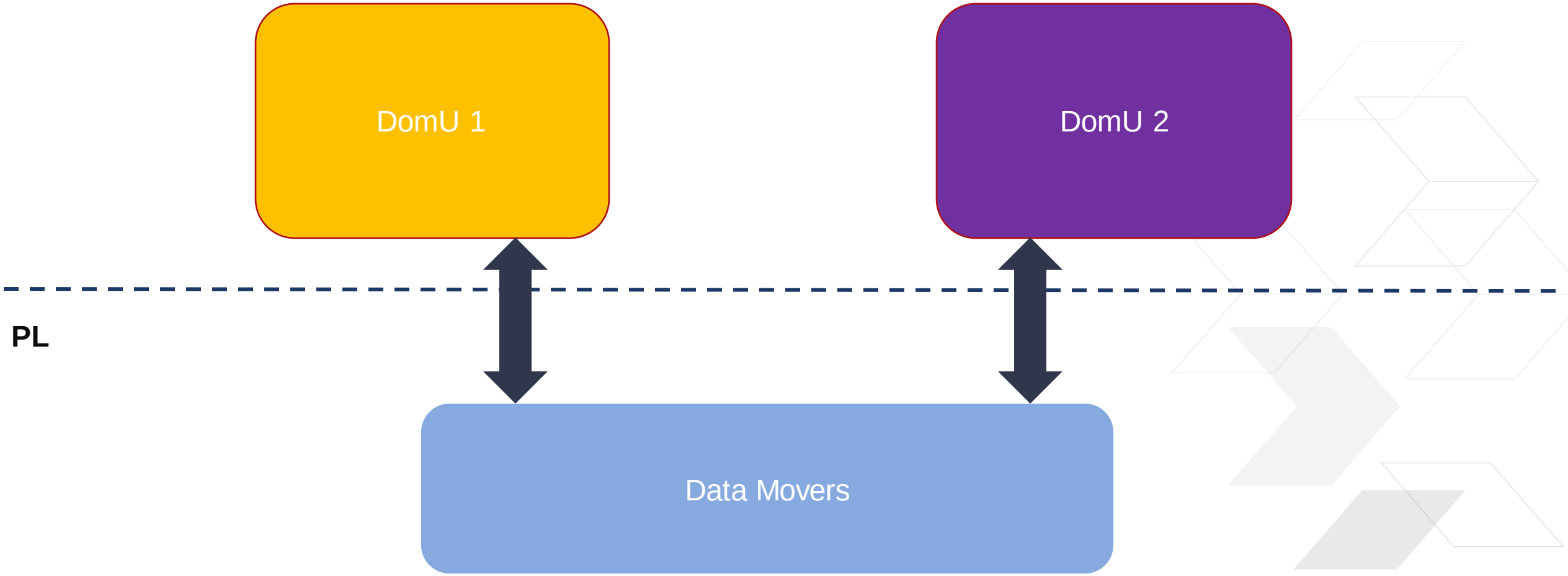
Static Shared Memory and Interrupts

- **Plain shared memory region**
  - Configured at "build time"
  - Guests setups ring buffers over shared memory
  - Can use OpenAMP RPMesg or any other communication libraries based on shared memory
- **Interrupt-based notifications, work with any OSes**
  - New hypercall to inject SGIs (patch by Xilinx)
- **Pros:**
  - Very simple
  - Works with any OS
  - Great performance if used correctly
- **Cons:**
  - One non-upstream patch to enable interrupt notifications
  - Require your own communication library
  - No dynamic connections

PL-based communication mechanisms



PL-based communication mechanisms



PL-based communication mechanisms

- **Create Data Movers in Programmable Logic**
  - From simple Network Devices to optimized Data Movers
- **Assign PL resources to VMs**
- **VMs use PL to send and receive data to/from other VMs**

- **Pros:**
  - Fastest for larger data sizes
  - Userspace drivers only
  - Easy to enable in any OS
- **Cons:**
  - Requires PL

Summary

Solution	Upstream Status for regular Xen	Upstream Status for Dom0less	VM-to-VM Communication vs. Device Virtualization	Compatibility	Performance	Unprivileged Backends
Plain shared memory & interrupts	Patch available for interrupts	Patch available for interrupts	Static VM-to-VM Communication	Can run anywhere	High if implemented correctly	Yes
Argo	Upstream	One patch for Linux available	Dynamic VM-to-VM Communication	Linux, Windows with a small effort	High	Yes
Xen PV Drivers	Upstream	Patches available soon	Unprivileged Device Virtualization	Most traditional OSes (Linux, Windows, BSDs)	High	Yes
VirtIO	Hypervisor: upstream Toolstack: patches available	No	Privileged Device Virtualization	Most traditional OSes (Linux, Windows, BSDs)	High with full privileged Otherwise: ?	No (work in progress)

Conclusions

- **Several solutions are already available, but nothing works out of the box yet**

- **No one-size fits all:**
  - Shared memory and notifications: best for OS compatibility
  - Argo: best for VM-to-VM communication
  - Xen PV Drivers: best for virtual devices with unprivileged backends
  - VirtIO: best for virtual device classes available

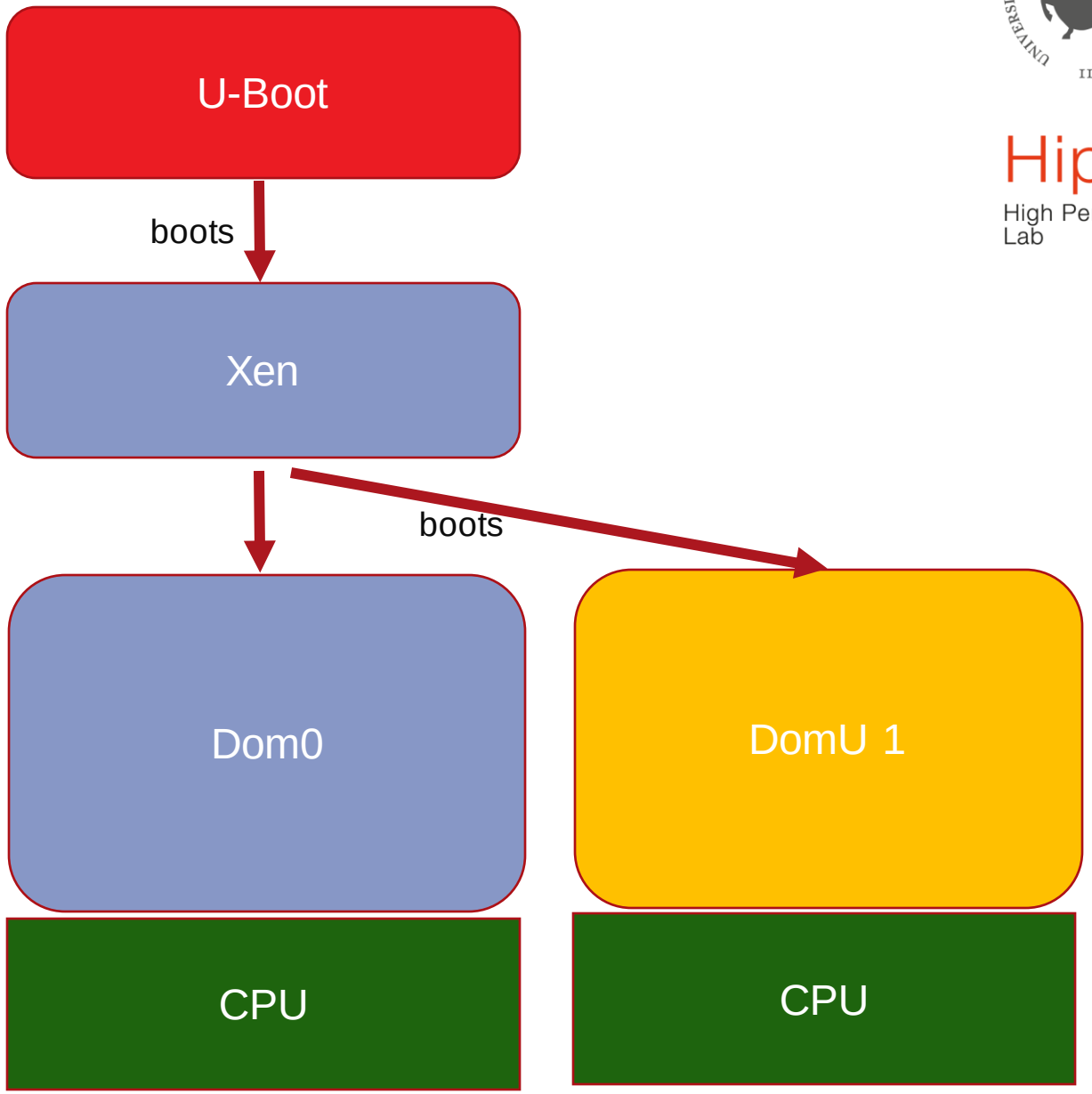
Demo

By Luca Miccio and Marco Solieri

Demo: Dom0less + PV Drivers

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

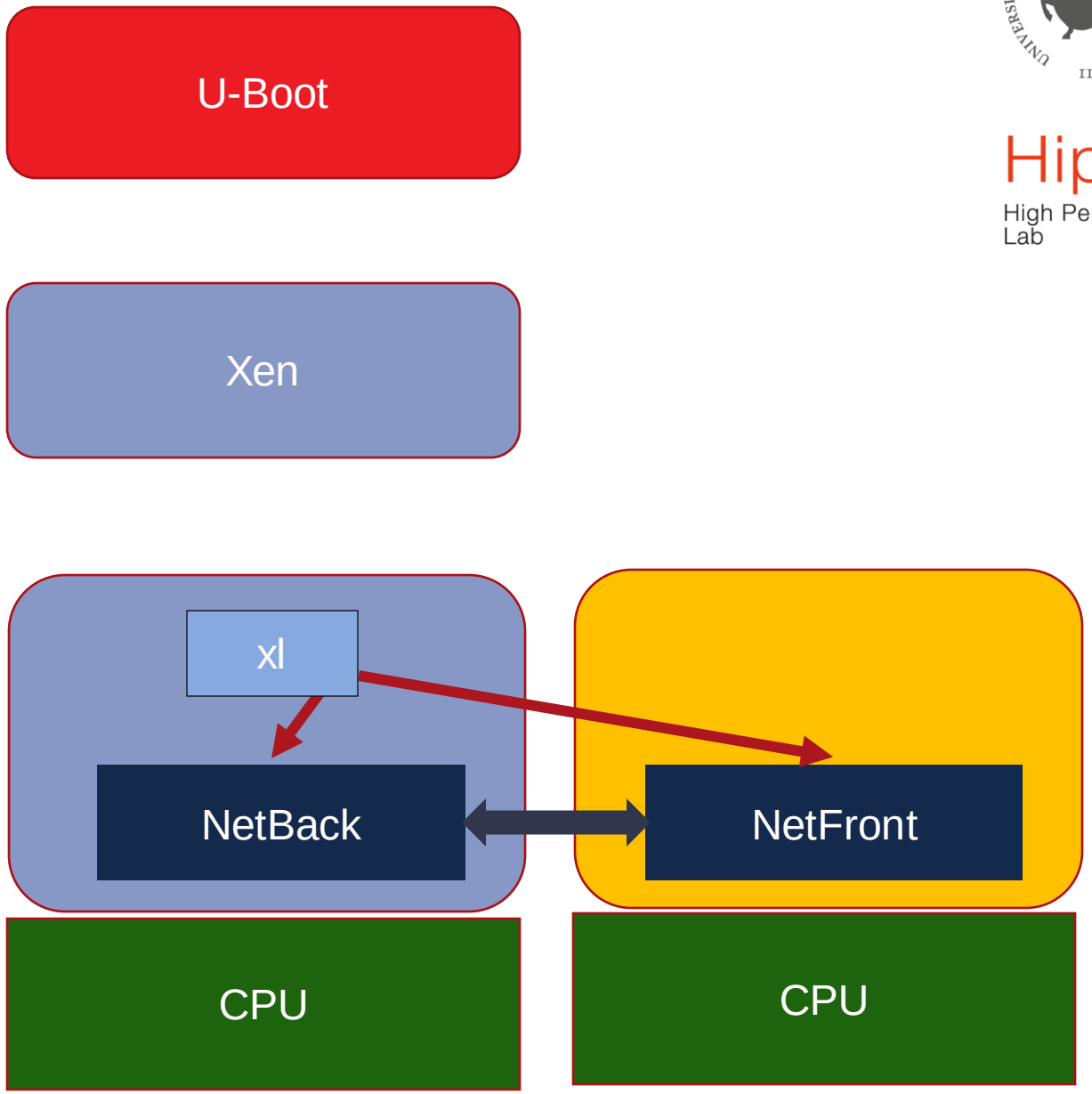
Hipert/Lab
High Performance Real Time Lab



Demo: Dom0less + PV Drivers

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Hipert/Lab
High Performance Real Time Lab



Adaptable.
Intelligent.